Konrad-Zuse-Zentrum
für Informationstechnik Berlin

Takustraße 7
D-14195 Berlin-Dahlem
Germany

SVEN BURGER, ROLAND KLOSE, ACHIM SCHÄDLE, AND
LIN ZSCHIEDRICH

HelmPole - A finite element solver for scattering problems on unbounded domains: Implementation based on PML

ZIB-Report 03-38 (November 2003)

Abstract

The solution of scattering problems described by the Helmholtz equation on unbounded domains is of importance for a wide variety of applications, for example in electromagnetics and acoustics. An implementation of a solver for scattering problems based on the programming language Matlab is introduced. The solver relies on the finite-element-method and on the perfectly-matched-layer-method, which allows for the simulation of scattering problems on complex geometries surrounded by inhomogeneous exterior domains. This report gives a number of detailed examples and can be understood as a user manual to the freely accessible code of the solver HelmPole.

Contents

Chapter 1

Introduction

`HelmPole` is a Matlab-program for the solution of scattering problems on unbounded domains in two dimensions, written by L. Zschiedrich [11]. The algorithm is based on the finite-element-method and the perfectly-matched-layer-method (PML method) originally introduced by Berénger, *cf.* [1]. In contrast to the interpretation of the PML, as an absorbing material, given by Berénger, the interpretation here is a complex continuation of the solution in the exterior domain. This ansatz can be found for example in the papers by Lassas and Somersalo [7, 6] and Collino and Monk [2] for homogenous exterior. Based on the work of F. Schmidt [8] `HelmPole` however allows for certain types of inhomogeneous exterior domains, including waveguide structures. A detailed description of the implementation can be found in [9]. Convergence issues of the PML are discussed in [4].

This manual gives a detailed description of `HelmPole`'s user-interface, which allows to adapt geometries and material properties as well as the characteristics of the incoming wave. The two main applications of `HelmPole` are electromagnetic and acoustic scattering.

Typical electromagnetic scattering problems are modeled by Maxwell's equations

$$\nabla \cdot \boldsymbol{B} = 0, \qquad \nabla \cdot \boldsymbol{D} = \rho, \tag{1.1}$$

$$\nabla \times \boldsymbol{E} = -\frac{\partial \boldsymbol{B}}{\partial t}, \qquad \nabla \times \boldsymbol{H} = \boldsymbol{j} + \frac{\partial \boldsymbol{D}}{\partial t}, \tag{1.2}$$

with permeability $\mu = 1$, vanishing charge density ρ, vanishing electric current density $\boldsymbol{j}$ and a spatial dependent relative permittivity $\epsilon(\boldsymbol{x})$. Assuming a harmonic time-dependency, equations (1.2) take the form

$$\begin{aligned} \nabla \times \boldsymbol{E}(\boldsymbol{x}) &= -\mu_0 i\omega \boldsymbol{H}(\boldsymbol{x}), \\ \nabla \times \boldsymbol{H}(\boldsymbol{x}) &= \epsilon_0 \epsilon(\boldsymbol{x}) i\omega \boldsymbol{E}(\boldsymbol{x}). \end{aligned} \tag{1.3}$$

Decoupling these equations yields the photonic wave equations

$$\nabla \times \left(\frac{1}{\epsilon(\boldsymbol{x})} \nabla \times \boldsymbol{H}(\boldsymbol{x}) \right) = \frac{\omega^2}{c^2} \boldsymbol{H}(\boldsymbol{x}), \tag{1.4}$$

$$\nabla \times (\nabla \times \boldsymbol{E}(\boldsymbol{x})) = \epsilon(\boldsymbol{x}) \frac{\omega^2}{c^2} \boldsymbol{E}(\boldsymbol{x}). \tag{1.5}$$

`HelmPole` assumes problems, homogeneous in z-direction. For light propagating only in xy-direction, it is possible to separate modes in two different polarizations. In the case of TE-polarization the $\boldsymbol{H}$-field takes the form $(0, 0, H_z)$, and the photonic wave equation (1.4) simplifies to

$$\nabla \cdot \left(\frac{1}{\epsilon(x,y)} \nabla H_z(x,y) \right) + \frac{\omega^2}{c^2} H_z(x,y) = 0. \tag{1.6}$$

In the case of TM-polarization the E-field takes the form $(0, 0, E_z)$, and the photonic wave equation (1.5) simplifies to

$$\triangle E_z(x,y) + \frac{\omega^2}{c^2} \epsilon(x,y) E_z(x,y) = 0. \tag{1.7}$$

For an introduction to acoustic scattering we refer to [5], where a derivation of the Helmholtz equation (1.8) from linearized equations for compressible fluids may be found. The stationary pressure p in a compressible fluid is given by

$$\Delta p + k^2 p = 0 \tag{1.8}$$

with the wavenumber

$$k := \frac{\omega}{c}, \tag{1.9}$$

where ω is the circular frequency of the wave and c is the speed of sound.

The plan of this manual is as follows. Chapter 2 describes the installation of the Matlab package `HelmPole.tar`. Chapter 3 guides the user through a test run of `HelmPole` with the parameters of the 'wave-guide'-example. In Chapter 4 several further examples give the user a feeling for the different problems which can be approached by `HelmPole`. Chapter 5 provides a detailed explanation of all variables which can be modified by the user. Chapter 6 gives a short introduction to the theory of the PML method and describes the coordinate system used to discretize the exterior. Chapter 7 shortly presents a conversion interface to the triangular grid generation tool `triangle` by J. Shewchuk.

Chapter 2

Installation

To install `HelmPole`, extract the tarred file `HelmPole.tar.gz` by typing:

```
tar -xzvf HelmPole.tar.gz
```

This creates a directory `/HelmPole/`, containing the subdirectories `/examples/`, `/fun/`, `/geo/`, `/trianglefiles/`, the function `startup.m`, and the file `main.dll`.

The subdirectory `/examples/` contains examples of problem description files (see chapter 5.2). User defined functions for the description of materials, boundaries, incoming fields, *etc.* are located in the subdirectory `/fun/`. The subdirectory `/geo/` contains geometry description files (see Section 5.3), which have been created using the files located in the subdirectory `/trianglefiles/` (see chapter 7).

To run `HelmPole`, you need the programming language Matlab (`http://www.mathworks.com`). For applications with complex, user-defined geometries, we recommend the triangulation tool `triangle` by J. Shewchuk, freely accessible under `http://www-2.cs.cmu.edu/~quake/triangle.html` (see chapter 7).

To check if your version is working, go through the example described in Chapter 3.

Chapter 3

Quick Start

In this chapter we go through a sample session of `HelmPole` step by step. This should give a basic idea of the program and its possibilities. A more complete description of the supported features is given in Chapter 5. In Chapter 4, a detailed description of various further examples is given.

In our example we simulate the propagation of light in an infinite straight waveguide in space $\mathbb{R}^2$. To obtain a bounded computational domain we introduce an artificial, so called 'outer' boundary. The computational domain we get this way in the example is a square region with a straight waveguide of high refractive index material and a low index background (see Figure 3.1). The light field reaching the computational domain from the exterior from left is the lowest mode of the waveguide.

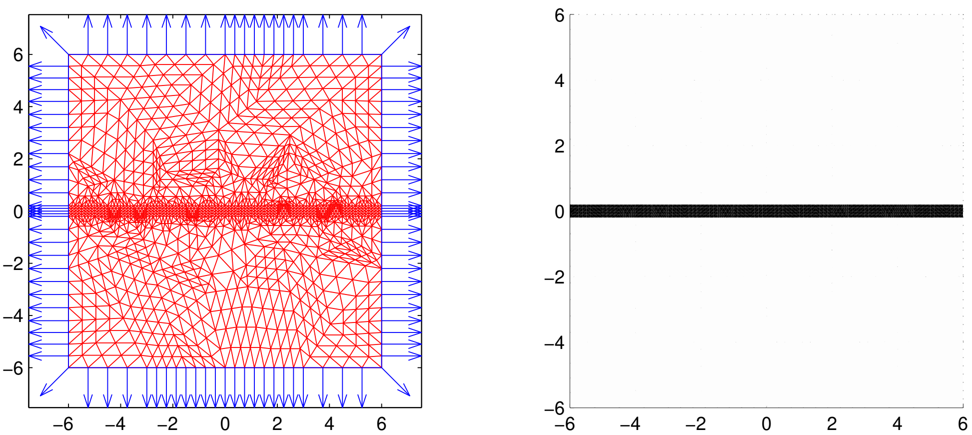

Figure 3.1: Geometry and material distribution of the problem `Waveguide.m`.

3.1 Getting started

Start Matlab and change to your `HelmPole` directory. Within this directory you locate the folders

`/examples/`, `/fun/`, `/geo/`, `/trianglefiles/`,

the function

`startup.m`,

and the file

`main.dll`.

Set the necessary environment variables by typing:

```
startup
```

Else you can start Matlab from the directory `HelmPole`. Matlab will then automatically set the necessary environment variables by examining the file `startup.m`

3.2 The problem description file: `Waveguide.m`

Start an editor and load the file `Waveguide.m` which is located in the directory `/examples/`. This file is the interface in which the user can choose from the different features of the program and set variables.
The file looks as follows:

```
function [problem, AlgPara] = waveguide(Para)
;
%% template for problem description file
%% for data structure and syntax read README
%% ------ geometry, material,sources and incoming waves -----
k0 = 1;
TM = 1;
CoarseGrid       = 'w';

Material.File    = 'Waveguide_material';
Material.Para.k  = k0;
Material.Para.n1 = 1.45; % relativ refraction index of substrate material
Material.Para.n2 = 6.6;  % relativ refraction index of waveguide material

IncWave.File     = 'Waveguide_income';
IncWave.Para     = Material.Para;
IncWave.Para.TM  = TM;
IncWave.Para.WaveGuidePos = [ -6 -6 ; ...
                             -0.2 0.2];
Exterior.Material.File   = 'Waveguide_material';
```

```
Exterior.Material.Para   = Material.Para;

Exterior.Ray.File        = 'makeRays_normal';
Exterior.Ray.Para        = [];

Exact.File = 'Waveguide_exact';
Exact.Para = IncWave.Para;
Exact.Para.lambda = 5.04973;

%% -------------- algorithmic parameters -------------------

Int.UniRefine = 0;
Int.PreRefine = 0;
Int.PpW = 5;

FEM.Type = 'quadratic';

TBC.Method = 'PML';
TBC.PML.k = k0;
TBC.PML.xi       =  [0 : 2^3].^3/2^9;
TBC.PML.DampFun =  'pml_jump';
TBC.PML.d        = 1;

%% ------------------------- Plots -------------------------

Plots.Triangulation = 1;
Plots.Material = 1;

Plots.Solution.Real = 1;
Plots.Solution.Imag = 0;
Plots.Solution.Abs = 1;
Plots.Solution.Log = 0;
Plots.Solution.Contour = 0;

Plots.ExactSolution.Real = 1;
Plots.ExactSolution.Imag = 0;
Plots.ExactSolution.Abs = 1;
Plots.ExactSolution.Log = 0;
Plots.ExactSolution.Contour = 0;
%% -----------  Do not edit beyond this line !! ------------
...
```

First, geometry, material, sources, and incoming waves are set: In the first lines the wavenumber `k0` is set, and the searched light field is chosen to be TM polarized. Then the string `'w'`, the name of the binary `.mat` file `w.mat` is specified as the name of the file with the geometry description of the computational domain. In this case our computational domain is a square, with a horizontal waveguide. For details of the geometry description see Section 5.3. The file `w.mat` is located in the folder `/geo/` and has been created previously by a triangulation tool (see Chapter 7).

Next, different refractive indices are attributed to different regions in the geometry of the interior domain by setting `Material.File = 'Waveguide_material'`, and the above defined wavenumber is attributed to the interior domain. For the syntax of the `Material.File` see Section 5.4. `Material.Para` is a structure that is passed to the `Material.File`. The file `Waveguide_material.m` is located in the directory `/fun/`.

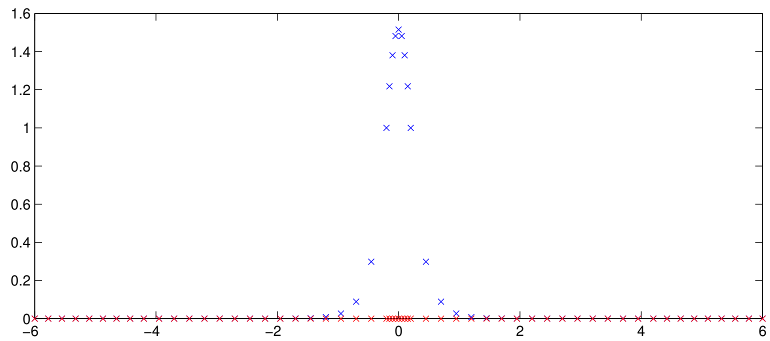

Figure 3.2: Amplitude distribution of the incoming wave along the left part of the outer boundary for the problem Waveguide.m.

With `IncWave.File = 'Waveguide_income';` the incident wave is specified. In the following lines parameters for the incoming field are set. These are the material parameters plus some additional parameters. The parameters are used by the file `Waveguide_income.m` in `/fun/` to calculate the Dirichlet data and the Neumann data at the left boundary of the computational domain. See Section 5.5 for a detailed description of the syntax of `IncWave.File`.

Then the exterior is specified: The exterior is divided into segments between rays starting at the computational domain and going outwards to infinity. `Exterior.Material.File` describes the refractive indices in the exterior. It has the same syntax as `Material.File` and in this case it is the same file. Thus we have to pass the parameters `Material.Para` to `Exterior.Material.Para`.

The `Exterior.Ray.File` specifies the discretization of the exterior. By specifying `makeRays_normal`, normal rays to the boundary are chosen. At corners, the normal is defined to be the mean of the two normal vectors to the adjacent

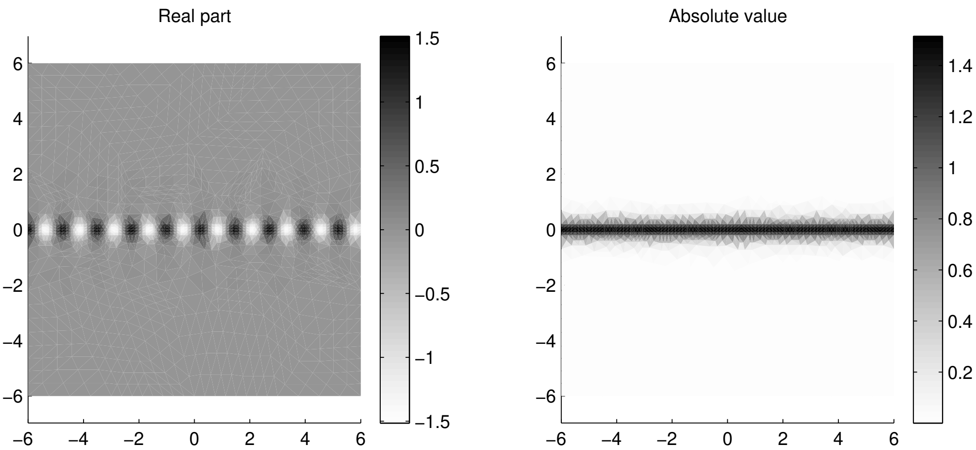

Figure 3.3: Real part and absolute value of the solution of the problem Waveguide.m.

edges. How to define rays in general is described in Section 5.6. In this case the structure `Exterior.Ray.Para` is empty, however, it cannot be omitted. The use of `makeRays_normal` should be limited to smooth computational domains. In the present example it gives a bad discretization of the exterior, because the outgoing segments at corners of the computational domain cannot be improved by choosing a finer computational grid in the interior. In the example described in Section 4.1 we will explain how to obtain "good" rays.

`Exact.File` is the name of the `.m` file located in `/fun/` that computes the exact solution for a given set of points. This function is supplied to the program in order to facilitate the study of the convergence of the discrete solution towards the exact solution. Parameters are passed to `Exact.File` with the structure `Exact.Para`. Except for the parameter `Para.la` these are the same as for the incoming wave. In our example `Para.la` is the first eigenmode of the waveguide. This eigenmode is calculated in `IncWave.File`.

Then algorithmic parameters follow: With `Int.UniRefine = 0;` we choose the number of uniform refinement steps of the coarse grid. Also, a pre-refinement that locally equalizes the number of discretization points per wavelength is omitted: `Int.PreRefine = 0;`. Instead, by setting `Int.PpW = 5` we refine the coarse grid locally such that there are at least 5 points per wavelength.

By `FEM.Type = 'quadratic';` , quadratic finite elements are chosen.

With `TBC.Method = 'PML';` the type of transparent boundary conditions is chosen to be the method of perfectly matched layers. `TBC.PML.k = k0` sets the wavenumber in the exterior domain. The discretization of each segment in the exterior domain is characterized by `TBC.PML.xi`. This is a vector giving the discretization of the variable ξ in (6.5). `TBC.PML.DampFun = 'pml_jump';` defines the damping function for the PML " `pml_jump` $= \gamma$ " *cf.* (6.6) and `TBC.PML.d`

`= 1;` sets the damping factor $\sigma = 1$ in (6.6). More details on how to set the algorithmic parameters are provided in Section 5.2. For a brief introduction to PML see Chapter 6.

In the following lines different features are chosen to be plotted during the program execution. Contour lines can only be plotted if the Matlab PDE toolbox is available. Some more lines follow in the problem description file which should not edited by the user and which are not described here.

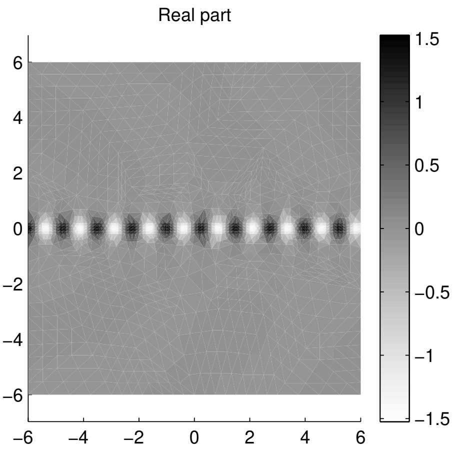

Figure 3.4: Real part and absolute value of the exact solution to problem Waveguide.m.

3.3 Running the program

Now we are ready to start the program by typing

```
main('Waveguide')
```

at the matlab prompt.

During the run, a number of windows opens to show some information to the user: The first opening window is shown in Fig. 3.1 The left part of this figure shows the geometrical description of the problem. The interior domain is triangulated, and the PML layer is divided into stripes, bordered by outgoing arrows. Obviously the exterior domain is not well resolved at the corners by the normal rays. A refinement of the interior domain would not improve that. The right part of the picture shows the distribution of the material refractive index in the interior domain.

The other opening windows show properties of the solution of the waveguide problem: Figure 3.2 shows the distribution of the incoming light field along the left boundary. Figure 3.3 shows the real part and the absolute value of the

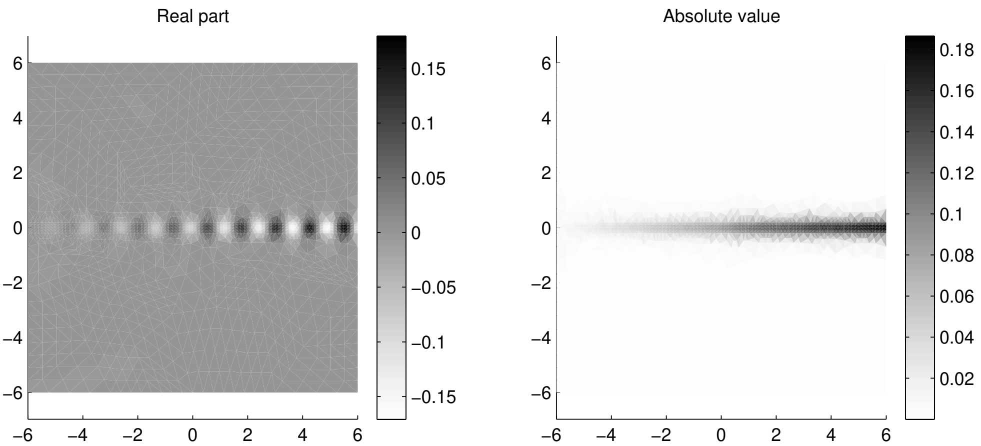

Figure 3.5: Real part and absolute value of the error of the solution (Fig. 3.3) of the problem Waveguide.m.

discrete solution. Figure 3.4 shows the exact solution of the problem, obtained semi-analytically, and Figure 3.5 shows the real part and the absolute value of the error of the discrete solution.

Additionally, Figure 3.5 shows the real part and the absolute value of the error of a more accurate solution, obtained from a run of `Waveguide.m`, using `Int.UniRefine = 1` and `Int.PpW = 10`.

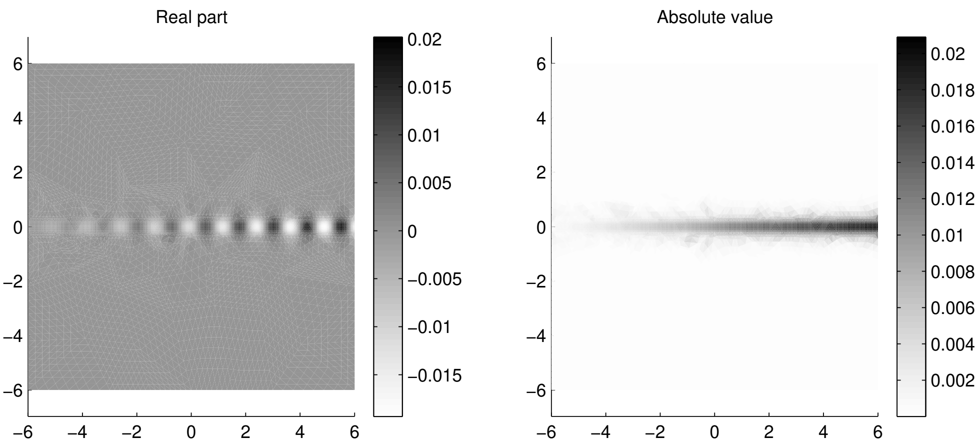

Figure 3.6: Real part and absolute value of the error of the problem Waveguide.m obtained with `Int.UniRefine = 1` and `Int.PpW = 10`.

On the command line some information is given on the progress of the main driver. If the exact solution is provided relative errors are printed.

Chapter 4

Examples

In this chapter we discuss some more examples. Section 4.1 shows how to describe waveguides in different geometries and how to obtain rays that give a better discretization of the exterior. Section 4.2 deals with the resonant coupling of optical waveguides via rectangular microcavities, with an example originally discussed in [3]. In Section 4.3 an example from acoustic scattering is discussed. We present two ways how to calculate the field scattered by a “soft” object. Section 4.4 discusses a Gauß light beam incident on the computational domain.

4.1 Waveguide - revisited

The example `Waveguide2a` is run by typing the command

```
main('Waveguide2a')
```

at the Matlab prompt.

In this case the computational domain is a rectangle given by $[0\,;\,1] \times [-2\,;\,2]$ with an infinite waveguide directed along an axis given by $y = 2x - 1$, *cf.* Figure 4.1.

Besides the different geometry of the problem, the main differences to the example `Waveguide` are the way how to describe the incident wave and the way how to define the rays for the discretization of the exterior. This is done by setting:

```
IncWave.File      = 'Waveguide_income2';
IncWave.Para      = Material.Para;
IncWave.Para.a    = -2;
IncWave.Para.b    = 1;
IncWave.Para.c    = 1;
IncWave.Para.d    = 0.4*sqrt(1/5);
IncWave.Para.Plot = 1;
```

and

```
Exterior.Ray.File = 'makeRays_waveguide';
Exterior.Ray.Para.vertices = [ 0 2;
                               1  2;
                               1 -2;
                               0 -2];
Exterior.Ray.Para.vray = [-1  2;
                           1  2;
                           1 -2;
                          -1 -2];
Exterior.Ray.Para.Waveguide = [0 -1.5 atan2(2,1) ;
                               0 -0.5 atan2(2,1) ;
                               1  1.5 atan2(2,1) ;
                               1  0.5 atan2(2,1)];
```

The incident wave is described using the function `Waveguide_income2` in subdirectory `/fun/`. This function can be used to describe the incident field for many different geometries. The parameters passed to this function are the wavenumber `Para.k`, the parameters `Para.a`, `Para.b`, and `Para.c` describing the axis of the waveguide $ax + by + c = 0$. The thickness of the waveguide is set by `Para.d`, see Fig 4.1. `Para.WaveGuideIndexin` and `Para.WaveGuideIndexout` are the relative refractive indices of the waveguide material and of the substrate material, respectively. `Waveguide_income2` calculates the incident wave at the "left" boundary of the computational domain which is the lowest mode of this waveguide. As we are calculating a guided wave, we can associate a direction with this wave. This wave is said to travel from "left" to "right". If we take the normal to the waveguide through the middle of the waveguide, this normal cuts the boundary of the computational domain in two pieces. The one that lies left to the normal is the "left" boundary of the computational domain. Prescribing the incident field only on the left boundary of the computational domain and setting is to zero for the right part of the boundary an error is introduced by the jump in the boundary condition. This error becomes visible if you chance the `Int.PpW` in `Waveguide2a` to 10 and it is prominent if you choose `Int.PpW` equal to 20. Choosing 20 instead of 10 points per wave length the maximum error decreases only negligible. Note that we discretize the PML according to the interior to see this behavior. The rays for the discretization of the exterior are calculated using the function `makerays_waveguide`. The rays have to be chosen such that the refractive index is constant in each segment of the exterior, i.e., between each pair of rays. Therefore the rays have to be adjusted to the waveguide in the exterior. In addition, the rays have to form an admissible discretization of the exterior, see Section 5.6 and Chapter 6. The parameters passed to `makerays_waveguide` are `Para.vertices`, `Para.vray` and `Para.Waveguide`. `Para.vertices` are the x, y coordinates of the polygonal computational domain in clockwise order. `Para.vray` are x, y coordinates of the rays $\vec{v}$ prescribed at the

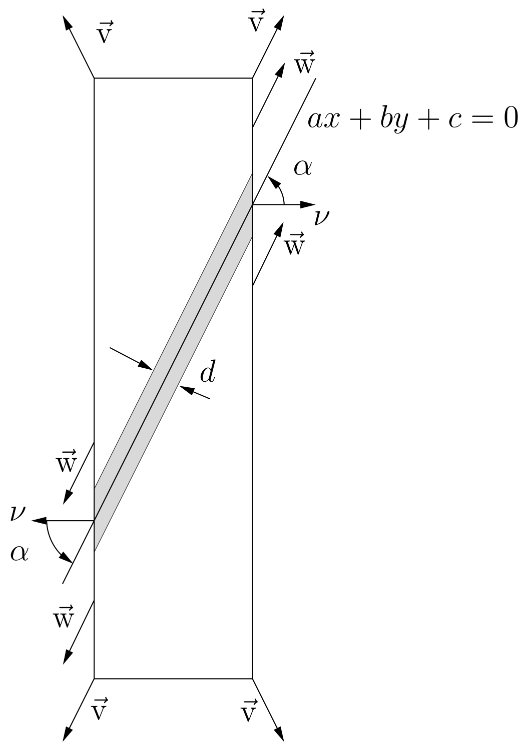

Figure 4.1: Geometry of waveguide in example `Waveguide2a`

edges, which have to be chosen to be admissible. `Para.vray` may be empty, in which case the rays are chosen to be the mean of the normal vectors of the two adjacent faces at each vertex. If we would not specify `Para.vray` in this example the rays at the vertices would intersect with the waveguide, and we could not obtain a valid discretization of the exterior. `Para.Waveguide` gives the position and direction of the waveguide. Each row of `Para.Waveguide` specifies one ray, by giving the x and y coordinate of the root point of the ray and the angle α between the ray vector $\vec{w}$ and the normal vector ν of the face of the polygon.

The geometry in example `Waveguide2a` seems to be very disadvantageous. The reason for that is not clear. Maybe the acute angle in the finite elements in the PML causes this problem.

Other examples with waveguides are `Waveguide2` and `Waveguide2b`.

4.2 Resonant coupling of optical waveguides

This example deals with the resonant coupling of optical waveguides via a rectangular microcavity (see Ref. [3]). To run the simulation invoke the script `mcr2analyser.m` which is located in the folder `/fun/`. In the problem description file `mcr2` the number of points per wavelength is set to 15, to obtain accurate results. However, this means that there are about 100.000 unknows in the linear system.

The coputational domain of the problem is shown in Figure 4.2. Two horizontal infinite straight waveguides are coupled by a rectangular microcavity. The outer boundary is again a rectangle. At port P_A we prescribe the incident wave traveling in the waveguide. At ports P_A, P_B, P_C and P_D the response of the microcavity is measured. We want to plot the power flux at the ports depending on the wavenumber, respectively wavelength of the incoming wave.

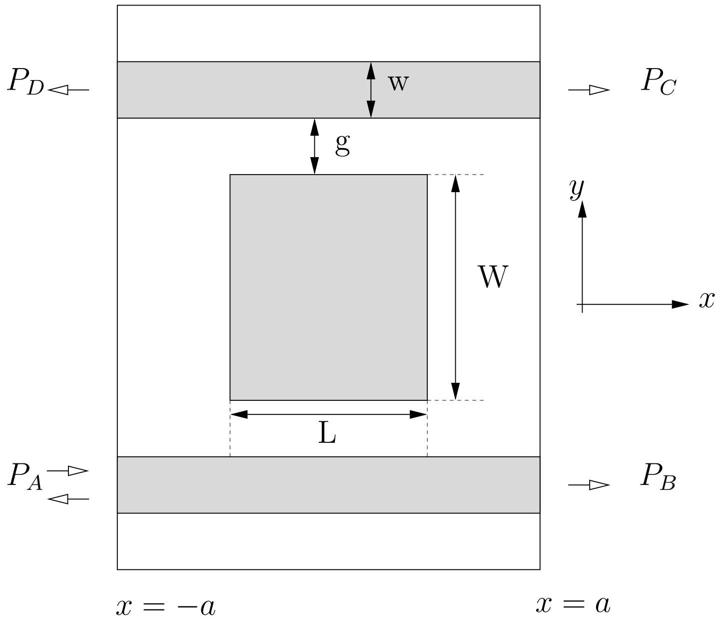

Figure 4.2: Resonance coupler

In this example we choose the parameters $W = L = 1.451$, $g = 0.4$ and $w = 0.073$. The waveguide material and the cavity have a relative refractive index of 3.4, and the substrate has relative refractive index of 1.45.

The problem description file is `mcr2.m`, located in `/examples/`. Let's have a look at it first. Using the argument `Para` we can pass different wavenumbers to the problem description file. Additionally, the first eigenmode of the waveguide and the relative refractive indices of the waveguide material and the substrate are passed to the problem description file. The function `mcr2_income` calculates the incident wave at port A. It is important that no further, "outgoing" fields are given anywhere. Using the function `makerays_waveguide` rays for the discretization of the exterior are calculated. See Section 4.1 for a detailed description of that function.

The script `mcr2analyser.m` in `/examples/` is essentially a loop over different wavelengths. In the first part the first eigenmode of the waveguide is calculated. As the two waveguides are of the same material and width w this is done once. With a call to `main` the problem is solved. The rest of the script analyses the scattered field and calculates the power flux along the waveguides at each port. The result is shown in Figure 4.3 .

Let's have a closer look at the physics of this problem. We start with some elementary considerations about guided waves for the Helmholtz equation $\Delta u(x, y)+k^2(y)\, u(x, y) = 0$ in a two dimensional space. Here the local wavenumber k does not depend on x. Recall that the electrical fields z-component of a TM-polarized electromagnetic wave satisfies a scalar Helmholtz equation. A guided wave is a field distribution which varies harmonically in x-direction,

$$u_g(x, y) = \psi(y)\, e^{ik_x x}.$$

Since u_g satisfies the Helmholtz equation we conclude that

$$\partial_{yy}\psi(y) + k^2(y)\,\psi(y) = k_x^2\psi(y)\,.$$

Hence ψ is an eigenfunction to the operator $\partial_{yy} + k^2(y)$. Using results from spectral theory we decompose a function $w(y)$ as

$$w(y) = \sum_l c_l\psi_l(y)\,,$$

where the ψ_l are eigenfunctions of the operator $\partial_{yy} + k^2(y)$. The coefficients c_l are given by the integral $c_l = \int_{-\infty}^{\infty}\psi_l^*(y)w(y)\,\mathrm{d}y$. Here our notation is formal since for the continuous part of the spectrum the sum must be replaced by an integral. The eigenfunctions ψ_l are orthogonal to each other,

$$\int_{-\infty}^{\infty}\psi_l^*\psi_l'\,\mathrm{d}y = \delta_{l,l'}. \tag{4.1}$$

For any splitting $c_l = c_{l,-} + c_{l,+}$ the function

$$u(x, y) = \sum_l c_{l,-}\psi_l(y)\, e^{-ik_{x,l}x} + \sum_l c_{l,+}\psi_l(y)\, e^{+ik_{x,l}x}$$

is a solution to the Helmholtz equation. The left sum corresponds to left traveling waves and the right sum to right traveling waves. To compute the splitting assume that also $v(y) = \partial_x u(x, y)$ is given for $x = 0$. Using the orthogonality relation we get $\int_{-\infty}^{\infty}\psi_l^* v(y)\,\mathrm{d}y = -ik_x(c_{l,+} - c_{l,-})$. Hence given any solution u to the Helmholtz equation $\Delta u(x, y) + k^2(y)\, u(x, y) = 0$ we are able to compute the above splitting by the two formulas

$$\begin{aligned} c_{l,-} + c_{l,+} &= \int_{-\infty}^{\infty}\psi_l^* u(0, y)\,\mathrm{d}y, \\ -ik_x(c_{l,+} - c_{l,-}) &= \int_{-\infty}^{\infty}\psi_l^*\partial_x u(0, y)\,\mathrm{d}y. \end{aligned}$$

We now consider the case of a TM-polarized time-harmonic electromagnetic wave in a waveguide. The electrical field is given as

$$E\left(x,y\right)=\left(\begin{array}{c}0\\0\\E_z\left(x,y\right)\end{array}\right)$$

The time average power flux along the waveguide direction is given by the x-component of the Poynting vector

$$S=\frac{1}{2}\Re\left(E\times H^*\right).$$

Making use of $H=-i\gamma\cdot\operatorname{curl}E$ with $\gamma=\sqrt{\mu\epsilon}/k$ we get

$$S_x=\frac{\gamma}{2}\Re\left(iE_z\partial_x E_z^*\right).$$

To compute the total power flux P_x in x-direction we integrate this expression over y. Splitting E_z as above,

$$E_z=\sum_l c_{l,-}\psi_l\left(y\right)e^{-ik_{x,l}x}+\sum_l c_{l,+}\psi_l\left(y\right)e^{+ik_{x,l}x}$$

and using the orthogonality relation (4.1) we end up with

$$P_x=\underbrace{\sum_l k_{x,l}\cdot c_{l,-}}_{P_{x,-}}-\underbrace{\sum_l k_{x,l}\cdot c_{l,+}}_{P_{x},+}.$$

Hence for each l in the sum above $P_{x,l,+}=k_{x,l}\cdot c_{l,+}$ is that part of the left going power flux $P_{x,+}$ which is transported by the lth guided mode. In the example above we want to compute the coupling of the scattered field with the fundamental modes of the waveguides. At each port the scattered field excites the fundamental mode of the waveguides. Let us restrict the analysis to Port C. Assume that the solution field $E_z\left(x,y\right)$ is given at every point $\left(x,y\right)\in\mathbb{R}^2$. Since there is no incoming wave from the right ($c_{0,-}=0$) the overlap integral

$$P_C=\int_{-\infty}^{\infty}E_z\left(a,y\right)\psi_0\left(y\right)dy. \tag{4.2}$$

is equal to $P_{x,0,+}$. The simulation only yields the total field E_z within the computational domain. But since the fundamental mode ψ_0 is evanescent outside the waveguide we replace the infinite integral in (4.2) by an integration over the right boundary of the computational domain. Figure 4.3 shows the power at the four ports vs. the wavelength of the incoming eigenmode. For the parameters, see the problem description file `mcr2.m` in `/examples/`.

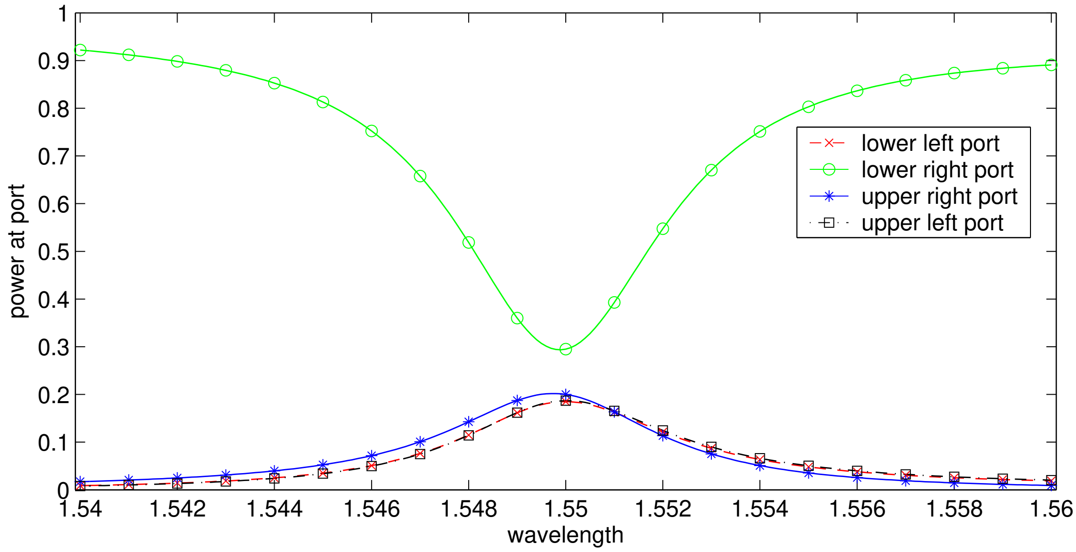

Figure 4.3: Normalized power at the four ports. A marker is plotted at every 10^{th} point.

4.3 Acoustic scattering

This example treats acoustic scattering by soft infinite cylindrical obstacles in $\mathbb{R}^3$. Problems of this type can be reduced to two dimensional scattering from closed curves.

Our computational domain Ω is a rectangular box with a hole. In order to have the analytic solution at hand we take a circular hole. The boundary of Ω naturally splits in two parts. One is the interior boundary Γ_{int}, the boundary of the hole. The other is the exterior boundary Γ_{ext}, where non-reflecting boundary conditions have to be imposed.

For a derivation of the Helmholtz equation (4.3) as a mean to model acoustic scattering see the introduction in the book by F. Ihlenburg [5].

$$\Delta p + k^2 p = 0 \ . \tag{4.3}$$

Here, p is the stationary wave and k is the wavenumber. The Helmholtz equation has to be supplemented by boundary conditions at infinity, i.e. radiation conditions for the scattered field. If we want to bound the computational domain we have to impose non-reflecting or absorbing boundary conditions at Γ_{ext}. This can formally be written using the linear operator DtN that given Dirichlet data on the boundary Γ for the Helmholtz equation in the exterior returns the Neumann data of the solution ($\partial_\nu p = DtN(p)$ on Γ_{ext}). One way to evaluate the operator DtN is the PML method. Additionally we have a Dirichlet boundary condition at Γ_{int}.

The total field p can be split into the known incident field p_{in} and the scattered field p_{sc}. All three are solutions of (4.3). At Γ_{int} we have that $p = 0 = p_{in} + p_{sc}$ and hence $p_{in} = -p_{sc}$.

Now there are two ways to formulate the problem:

- The first one is to derive an equation for the scattered field p_{sc}: p_{sc} has to be a solution of the Helmholtz equation,

$$\Delta p_{sc} + k^2 p_{sc} = 0. \tag{4.4}$$

  It has to fulfil the Dirichlet condition (soft scattering) $p_{sc} = g$ on Γ_{int}, where $g = -p_{in}|_{\Gamma_{int}}$, and it has to fulfil the Robin condition $\partial_\nu p_{sc} = DtN(p_{sc})$ at Γ_{ext} .

- The other way is to derive an equation for the total field p: p has to satisfy Equation (4.3). At Γ_{int} we have $p = 0$, and at the boundary Γ_{ext} the scattered field p_{sc} has to satisfy $p_{sc} = p - p_{in}$ $\partial_\nu(p - p_{in}) = DtN(p - p_{in})$.

Two examples for the formulations are provided. Both model the scattering of a plane sound wave by a soft circle. Try the examples `Cylinder.m` and `Cylinder2.m`.

In `Cylinder` the incoming wave is set to zero, the Dirichlet values on the interior boundary are set to the values of an incoming wave at these points. The exact solution is calculated with a slightly modified formula taken from [5], *cf.* p. 29-31, Example 2.4. A uniform refinement of the grid decreases the error by a factor of 1/2 (linear finite elements) or 1/4 (quadratic finite elements), respectively.

In Example `Cylinder2` the incoming wave is specified by prescribing Dirichlet and generalized Neumann data at the boundary of the computational domain Γ_{ext}. At the interior boundary homogenous Dirichlet data is set. In this case for quadratic finite elements a uniform refinement of the grid decreases the error only in the first refinement. The reason for this is that the circular geometry is not resolved by successive uniform refinements of the coarse mesh. To decrease the error one has to remesh the geometry.

4.4 Gauß beam

This examples calculates the field of a Gauß beam in a homogenous material. To run it type (you will need about 1.5 GB memory to run this example):

```
Para.k = 4;
Para.matb = 3.5;
Para.matg = 3.5;
main('gauss',Para);
```

According to the book by Unger [10], p. 304, the field of a Gauß beam is given by

$$E(r,z) = E_0 \frac{w_0}{w(z)} \exp\left(-\frac{r^2}{w^2(z)} - i\left(kz + \frac{\pi r^2}{\lambda R(z)} - \arctan\left(\frac{\lambda z}{\pi w_0^2}\right)\right)\right) \quad (4.5)$$

where $\lambda = 2\pi/k$ is the wavelength,

$$R(z) = z + \frac{w_0^4 k^2}{4z} \quad (4.6)$$

and

$$w(z) = \sqrt{w_0^2 + \frac{4z^2}{w_0^2 k^2}}\ . \quad (4.7)$$

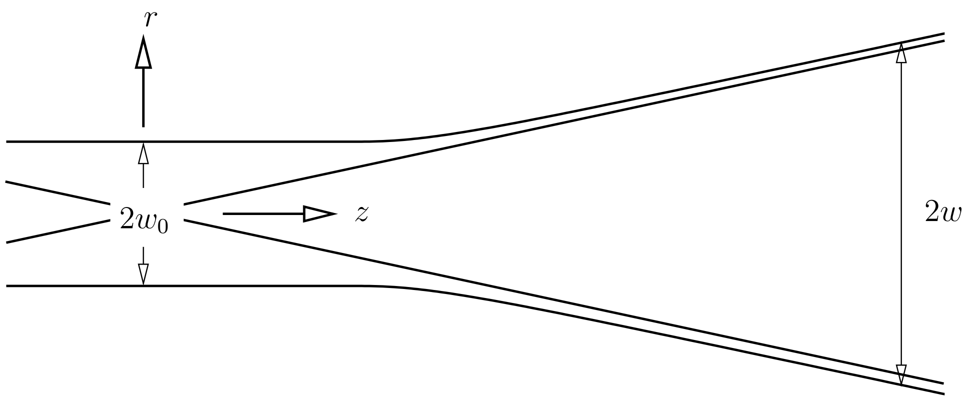

Figure 4.4: Gauß beam

The focus is at the point $r = 0, z = 0$. In the problem description file the parameters for the Gauß beam are

```
IncWave.Para.k   = Material.Para.matb*k0;
IncWave.Para.a   = 1;
IncWave.Para.b   = 0.5;
IncWave.Para.c   = 0;
IncWave.Para.w0  = 1;
```

This means in x, y coordinates the main axis is given by $ax + by + c = 0$ and the focus is the cut of the main axis with the orthogonal axis $bx - ay + c = 0$. The "exact" solution is the Gauß beam in the computational domain, which is only an approximate solution to the Helmholtz equation. Thus the error is mainly due to modeling a wave by a Gauß beam.

Typing

```
Para.k = 4;
Para.matb = 3.5;
Para.matg = 1;
main('gauss',Para);
```

calculates a Gauß beam as before incident on a photonic crystal with 3×7 holes on a hexagonal grid. The error calculated for this setting of the material parameters is meaningless.

Chapter 5

Function References

5.1 The main program: `MAIN`

Syntax

```
[PROBLEM,SOLUTION,ERROR,BOUNDARY] = MAIN(PROBLEMNAME)
[PROBLEM,SOLUTION,ERROR,BOUNDARY] = MAIN(PROBLEMNAME,PARA)
```

Argument

- `PROBLEMNAME` is a string, specifying the name of a MATLAB function that describes the equation to be solved, the geometry, and the transparent boundary condition, *cf.* 5.2. Some example files with problem descriptions are included in `/examples/`. An example template is given in `/example/template`.
- `PARA` is structure, it allows to pass any parameters to the main program.

Description

- `[PROBLEM,SOLUTION,ERROR,BOUNDARY] = MAIN(PROBLEMNAME)` returns four structures:
  - `PROBLEM` describing the problem. The most important substructure is `PROBLEM.GEO` with the fields `P`, the point data of the triangulation, `T`, the triangle data of the triangulation and `E` the edge data. All three are matrices as described in Section 5.3.
  - `SOLUTION` a structure with the fields `SOLUTION.U` `SOLUTION.UL` and `SOLUTION.UG` representing the solution. `U` is the solution-vector in the interior domain and `UL` is the solution restricted to points `PROBLEM.GEO.P`. If you use linear finite elements `UL` and `U` will be the same. `UG` is the solution-vector of the complete system, including the

PML layer. If `nn` is the number of degrees of freedom in the interior domain `UG(1:nn)` equals to `U`. Setting the number of degrees of freedom in the PML layer equal to `npml`, `UG(nn+1:nn+npml)` is the solution in the PML layer. With the number of degrees of freedom on the outer boundary, `nbn`, the generalized Neumann data of the scattered field on the boundary is `UG(nn+npml+1:nn+npml+nbn)`.

- `ERROR` If `EXACT.FILE` in the problem description file is empty, `ERROR` is empty, too. Else, `ERROR` has four fields: a) `ERROR.FIELD`, a vector of the same size as `SOLUTION.U` giving the pointwise error in the interior, b) `ERROR.L2,` the relative l_2 error, c) `ERROR.LINF`, the relative error in the maximum norm and d) `ERROR.REF` the exact solution in points `PROBLEM.GEO.P`.
- `BOUNDARY` a structure describing the boundary. We will not describe it in detail here. `BOUNDARY.P` are xy coordinates of the boundayr points of the triangulation. `BOUNDARY.RAY` are the rays used to discretize the exterior. The transpose of `BOUNDARY.PMP` is the projection for function values given on points of the triangulation, `POEBLEM.GEO.P`, onto boundary points, `BOUNDARY.P`.

Example

```
[p,s,e,b]=main('Waveguide');
```

calls the example Waveguide, described in Section 3. There quadratic finite elements are used. With

```
quiver(b.p(1,:),b.p(2,:),b.ray(1,:),b.ray(2,:))
```

you can plot the rays.

```
plot3(b.p(1,:),b.p(2,:),abs(b.PMP'*s.ul),'x'); hold on
```

plots the absolute value of the “linear” part of solution on the boundary.

5.2 User interface: `PROBLEMNAME`

The problem description file is located in the directory `/HelmPole/`. Examples of problem description files can be found in `/examples/`. It is the the main user interface where the parameters of the program run are set.

Syntax

`[PROBLEM,ALGPARA] = PROBLEMNAME(PARA)`

Argument

- `PARA` structure, this is the structure of parameters passed to `MAIN`.

Description

`PROBLEMNAME` returns two structures. `PROBLEM` describes the problem parameters and `ALGPARA` describes the algorithmic parameters. The options for the different parameter choices are listed below.

- `k0`: real, wavenumber
  The wavenumber `k0` specifies the wavenumber in equations (1.7) and (1.6).

- `TM`: integer, equation type
  The `TM`-switch specifies which equation is to be solved:
  `TM=1`: corresponds to equation (1.7) (TM polarization),
  `TM=0`: corresponds to equation (1.6) (TE polarization).

- `CoarseGrid`: .mat coarse grid file (string), geometry description
  The variable `CoarseGrid` is a string specifying the name of the `.mat` file with the description of the geometry of the problem. The file is located in the directory `/geo/`. The contents of this file is described in Section 5.3.

- `Material`: This specifies the nature of the materials in the interior computational domain.

  – `Material.File`: .m file (string), local wavenumber
    `Material.File` specifies the name of the corresponding file located in the directory `/fun/`. The contents of this file is described in section 5.4.

  – `Material.Para`: structure, parameters for `Material.File`. `Material.Para` may be empty.

- `IncWave`: This specifies the incident wave.

- `IncWave.File`: .m file (string), incident wave on the outer boundary `IncWave.File` specifies the name of the file with the description of the incident field. The file is located in the directory `/fun/`.
- `IncWave.Para`: structure, parameters for `IncWave.File`. `IncWave-.Para` may be empty.

- `Exterior`: structure to describe the exterior.
  - `Exterior.Material.File`: .m file (string), local wavenumber for the exterior
    `Exterior.Material.File` specifies the name of the file with the description of the material distribution in the exterior. The wavenumber is constant in each segment described by the rays. This allows to simulate waveguides.
  - `Exterior.Material.Para`: structure, parameters for `Exterior.Material.File`, may be empty.
  - `Exterior.Ray.File`: .m file (string), a function describing the rays of the exterior discretization, see Section 5.6. There are three files already provided in `/fun/`, namely `makeRays_normal.m`, `makeRays_radial.m` and `makeRays_waveguide.m`.
  - `Exterior.Ray.Para`: structure, parameters passed to `Exterior.Ray-.File`. The structure `Exterior.Ray.Para` may be empty.

- `Dirichlet`: structure, Dirichlet boundary data
  This structure may be omitted if there are no interior edges, *cf.* Section 5.3.
  - `Dirichlet.file`: .m file (string), Dirichlet data on interior boundaries. `Dirichlet.file` is the name of the file that calculates the Dirichlet boundary values on the interior boundary.
  - `Dirichlet.Para`: structure, parameters passed to `Dirichlet.file`, may be empty.

- `Exact.File`: .m file (string), name of the file to calculate the exact solution in the interior. The parameters passed to `Exact.File` are `Exact.Para`.

- `Int.UniRefine`: integer, number of uniform refinement steps of the coarse grid (useful for convergence-studies).

- `Int.PreRefine`: integer, number of local refinement steps to obtain a 'balanced' grid, i.e., the grid is refined where the local wavenumber is high (obsolete, use `Int.PpW` instead). The algorithm picks the triangles with the top 20 % of the largest triangles weighted by the local wavenumber.

- `Int.PpW`: real, local refinement to obtain at least `Int.PpW` discretization points per wavelength locally.

- `FEM.Type`: string, finite element type
  `FEM.Type` is `'linear'` for linear finite elements and `'quadratic'` for quadratic finite elements.

- `TBC.Method`: string, specifies the method of the transparent boundary condition. The only method presently available in HelmPole is `'PML'` the perfectly matched layer.

- `TBC.PML`: structure, parameters for the PML.

  - `TBC.PML.k`: local wavenumber in each segment described by rays

  - `TBC.PML.xi`: vector, discretization along the rays
    `TBC.PML.xi` has to be sorted in ascending order starting from 0. The last value gives the thickness of the PML. `TBC.PML.xi` corresponds to the ray-like variable ξ in (6.5) As the solution inside the PML decreases exponentially, one does not gain much accuracy of the solution by increasing the PML largely.
    The distribution of discretization points inside the PML is arbitrary. The distance of two points in the PML close to the boundary should be chosen at the same order, maybe a factor of 2 less, as the distance of two points in the interior domain. A linear distribution will work well in most cases. It was observed that a cubic distribution often gives better results. That means there are less discretization points required at the end of the PML.
    In most cases it is sufficient to have around twenty points along one ray in your PML. However, if you observe reflections at the outer boundary, you should use more points.

  - `TBC.PML.DampFun`: .m file (string)
    The damping function is the function γ introduced in (6.6)

  - `TBC.PML.d`: real, damping factor for the PML
    It is save to set this factor to 1. `TBC.PML.d` equals σ in (6.6)

- `Plots`: structure, specifies which figure to open and what to plot.

  - `Plots.Triangulation`: boolean, plot the triangulation

  - `Plots.Material`: boolean, plot the local wavenumber

  - `Plots.Solution`: structure, specifies which features of the solution to plot

    * `Plots.Solution.Real`: boolean, plot the real part of the solution

  * `Plots.Solution.Imag`: boolean, plot the imaginary part of the solution
  * `Plots.Solution.Abs`: boolean, plot the absolute value of the solution
  * `Plots.Solution.Log`: boolean, plot the logarithm of the absolute value of the solution
  * `Plots.Solution.Contour`: boolean, plot contour lines
    Contour lines are only plotted if the Matlab PDE toolbox is available.

- `Plots.ExactSolution`: structure, specifies what features of the exact solution to plot, if it is available. It also plots the pointwise error. The usage is the same as for `Plots.Solution`

Example

See the example problem description files in `/examples/`.

5.3 Geometry description: `COARSEGRID`

The geometry description file stores the triangulation of the computational domain and specifies which of the boundaries are outer transparent boundaries and which are interior boundaries, where Dirichlet data is given.

Syntax

`CoarseGrid`

Description

`CoarseGrid` .mat file contains the arrays `p`, `e`, `t`, `outBound` and `intBound`.

- `p`: Points, $2 \times np$ matrix containing the Euclidian coordinates of the points, where np is the number of points.
- `e`: Edges, $4 \times ne$ matrix containing the boundary edges of the coarse grid, where ne is the number of edges. Each column of `e` corresponds to one edge:
  - `e(1, :)` index of start point
  - `e(2, :)` index of end point
  - `e(3, :)` edge marker to determine whether the edge is belonging to the inner or to the outer boundary
  - `e(4, :)` material index used by the file `Ext.Material.File`

  The computational domain lies on the left of an edge, i.e., all edges are orientated in counter-clockwise order.
- `t`: Triangles, $4 \times nt$ matrix containing the triangles, where `nt` is the number of triangles.
  - `t(1, :)` index of first vertex
  - `t(2, :)` index of second vertex
  - `t(3, :)` index of third vertex
  - `t(4, :)` material marker to be used by the file `Material.File`

  For the orientation see Figure 5.3.
- `outBound`: vector of edge markers indicating that the edge is belonging to the outer boundary. **Caution:** Please note that the outer boundary has to be a closed curve[1]. The computational domain, bounded by the outer

[1] In future releases this restriction will be overcome

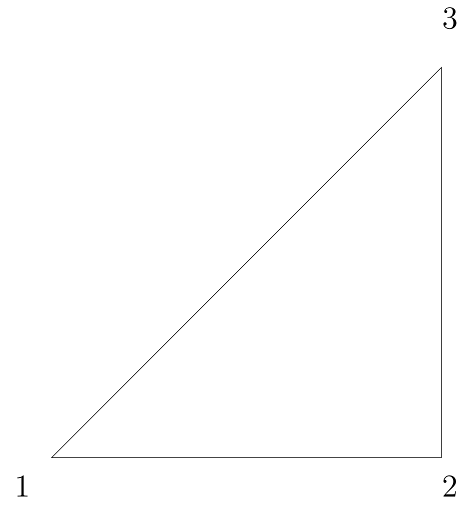

Figure 5.1: Orientation of the degrees of freedom on a triangle.

boundary, has to be starshaped. However for starshaped non convex domains great care has to be taken to chose a physically meaningfull problem. Hence the user is adviced to use convex computational domains.

- `intBound`: vector of edge markers indicating that the edge is an interior boundary. **Caution:** There are two restrictions regarding the inner boundaries.[2]:
  - The inner boundaries have to be closed curves as well.
  - At the inner boundaries only Dirirchlet boundary conditons can be set.

Example

By typing

```
load rect21;
pdemesh(p,e,t);
```

at the matlab prompt the most simple example of a triangulation of a rectangle with four points and two triangles is loaded to the workspace and plotted. Other example files are located in `/geo/`.

[2]We plan to overcome these restrictions in future releases

5.4 Material description: MATERIALFILE

The local properties of the material are specified in MATERIALFILE.

Syntax

`K = MATERIALFILE(MATERIALINDEX, PARA)`

Arguments

- MATERIALINDEX: vector
  MATERIALINDEX is the material index specified in `t(4,:)` and `e(4,:)`.
- PARA: structure, parameters passed to MATERIALFILE

Description

`K = MATERIALFILE(MATERIALINDEX, PARA)` returns the local wavenumber K for a triangle or edge, corresponding to the material with index MATERIALINDEX. PARA is a structure of auxiliary parameters.

The same syntax is used for the interior material description (`Material.File`), for the material description of the boundary (`IncWave-.Material.File`), and for the material description of the exterior (`Ext.Material.File`).

Example

Waveguides can be described with the following material file.

```
function k = waveguide_material(index, Para)

k0   = Para.k;
matb = 1.45;
matg = 6.6;

k = ones(1, length(index))*matg;
it1 = find((index==1) | (index==3));
k(it1) = matb;
k = k*k0;
```

With this, a relative refractive index of 1.45 is attributed to triangles and edges with material indices 1 or 3. The rel. refractive index of 6.6 is attributed to all others.

5.5 Incident wave file: INCWAVE

Calculates the Dirichlet data and generalized Neumann data on the outer boundary, generated by an incoming wave.

Syntax

`[U,DU] = INCWAVEFILE(P,RAY,PARA)`

Arguments

- `P`: $2 \times$ `nedof` matrix, boundary points
  (x,y) coordinates of points on the boundary, where `nedof` is the number of degrees of freedom – which in case of quadratic finite elements is `2*ne`.

- `RAY`: $2 \times$ `nedof` matrix,
  rays discretizing the exterior, see Section 5.6. Each column represents one ray. The rays are **not** normalized.

- `PARA`: structure, parameters that are passed to `INCWAVEFILE.`

Description

`[U,DU] = INCWAVEFILE(P,RAY,PARA)` calculates the Dirichlet values, `U`, and the directional derivative of the incoming wave in direction of the rays, `DU`, on the degrees of freedom on the outer boundary. $\mathtt{DU} = (\nabla \mathtt{U}, \mathtt{RAY})$. `U` and `DU` are column vectors.

Example

Consider a plane wave, $\exp(ikx)$, given boundary points `p`, rays `ray`, and the parameters of the structure `IncWave.Para` with fields `k` and `angle`. Then, `EW_income` calculates the Dirichlet values `u` of the incoming wave and the generalised Neumann data `us`, $us = (\nabla u, ray)$ on the boundary.

```
function [u,us] = EW_income(p,ray,Para)

k     = Para.k;
alpha = Para.angle;
u = [exp(i*k*(cos(alpha)*p(1,:)+sin(alpha)*p(2,:) ))];
dxu=[i*k*cos(alpha)*exp(i*k*(cos(alpha)*p(1,:)+sin(alpha)*p(2,:)))];
dyu=[i*k*sin(alpha)*exp(i*k*(cos(alpha)*p(1,:)+sin(alpha)*p(2,:)))];

u = u.';
us = sum( [ dxu ; dyu ] .* ray).';
```

5.6 Discretization of the exterior: `MAKERAYS`

`MAKERAYS` is used to calculate the rays for the discretization of the exterior.

Syntax

`RAY = MAKERAYS(P,S,PARA)`

Arguments

- `P`: $2 \times$ `nbp` matrix of (x,y) coordinates of boundary points. `nbp` is the number of boundary points.
- `S`: structure, describing the segments
  `S.p_i` is a $2 \times$ `nbp` matrix of point indices of vertices on the boundary.
- `PARA`: structure, any additional parameters for `MAKERAYS`

Description

`RAY = MAKERAYS(P,S,PARA)` calculates a $2 \times$ `nbp` array of rays. Each column represents one ray. The rays are normalized to euclidian length 1.
The rays have to form an admissible discretization of the exterior. To put it simple: The discretization of the computational domain transforms the boundary of the computational domain to a polygon (faces). The exterior is modelled by quadrilateral elements, each defined by a face of the computational domain and by the two rays starting in the adjacent vertices of the face and going outwards. If you start drawing a parallel to one face, this has to intersect the two rays. From these intersections one draws again parallels to the adjacent faces and so on. The rays are admissible if in the end the last two parallels and the last ray intersect in one point.
Formally this means that the mapping Q given in Equation (6.11) is continuous.

There are three general purpose methods supplied in `/fun/`.

- `makerays_radial` calculates radial rays from an origin given by the vector `PARA.LineCenter`. This gives very good discretizations for circles and squares, if `LineCenter` is chosen to be the center of the circle or square. It also works for rectangles where the ratio of the two different sides is not too large. See left of Fig. 5.2, in this examples, the center of the rays is the center of the rectangle.
- `makerays_normal` calculates normal rays to the boundary. This works well as long as the boundary does not have too sharp corners. In this case `PARA` is empty. See right of Fig. 5.2. At corners the normal direction is the average of the normals of the two adjacent faces. `makerays_normal` can lead to very

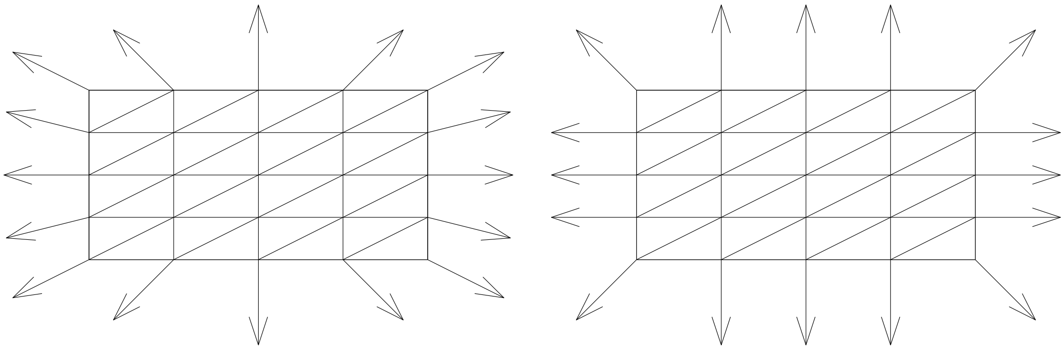

Figure 5.2: Radial (left) and normal (right) rays for the exterior surrounding a rectangular domain.

bad discretizations of the exterior, as a refinement of the interior will not improve the approximation of the exterior at the corners.

- `makerays_waveguide` is an elaborate tool to calculate rays for polygonal domains. It allows to calculate rays that fit waveguides in the exterior. The parameters are listed below:
  - `vertices`: `nv` $\times 2$ matrix containing the x, y coordinates of the vertices of the polygon in clockwise order. Here `nv` is the number of vertices.
  - `vray`: `nv` $\times 2$ matrix containing the x, y coordinates of the rays starting a the vertices of the polygon. They have to be chosen in accordance with the rule given above. In case `vray` is empty "normal" rays at vertices are chosen, i.e., the rays are the mean value of the normals of the two adjacent segments. `vray` are the arrows in the corners of the rectangle shown in the left of Figure 5.3 .
  - `Waveguide`: `nwr` $\times 3$ matrix of the rays describing the waveguide. Each row in `Waveguide` specifies the x, y coordinates of the starting points of a ray and the angle of this ray with the normal to the face of the polygon on which this starting point lies. Each of the `nwr`/2 waveguides is specified by its two enclosing rays. In the example shown in Figure 5.3 four parallel vectors are given, indicating the positions and the directions of the waveguides entering the computational domain on the left and on the right. In this case the waveguides are continued horizontally through the computational domain.

The function `makerays_waveguide` calculates all other rays for the discretization of the exterior from the rays specified in `vertices`, `vray`, and `Waveguide`. It uses weighted averages of the given rays for the determination of the remaining rays. For the specified rays shown in the left of

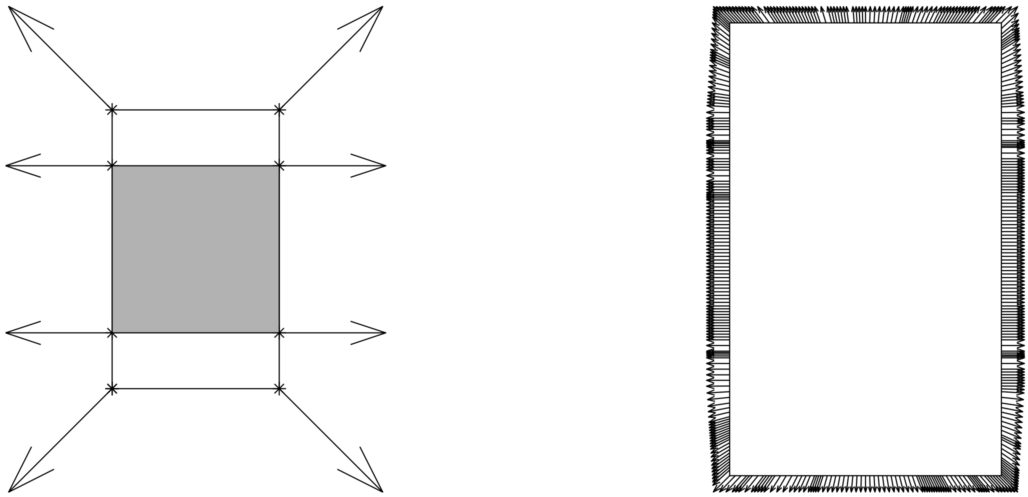

Figure 5.3: Exterior rays for a problem with waveguides entering the computational domain.

Figure 5.3 the whole set of exterior rays is shown in the right part of the same Figure. Please see also the example described in Section 4.1.

Caution: `RAY` is not the variable that is passed internally to `INCWAVE`.

5.7 Dirichlet data

Setting Dirichlet data at interior boundaries.

Syntax

```
[U] = DIRICHLETFILE(P,PARA)
```

Arguments

- `P`: matrix, (x,y) coordinates of points on the interior boundary.
- `PARA`: structure, parameters passed to `DIRICHLETFILE`

Description

`[U] = DIRICHLETFILE(P,PARA)` returns the Dirichlet data `U` on points `P` of the interior boundary. Please note: It is not sufficient for quadratic finite elements to return the Dirichlet values only on the grid points of the interior boundary. Rather `DIRICHLETFILE` has to return the Dirichlet data in all degrees of freedom on the interior boundary.

5.8 Exact Solution

Function to calculate the exact solution pointwise on a grid.

Syntax

```
U = EXAKTFILE(P,PARA)
```

Arguments

- `P`: $2 \times$ `np` matrix,
  (x, y) coordinates of points (degrees of freedom) in the interior of the computational domain.
- `PARA`: structure, parameters passed to `EXAKTFILE`

Description

`U = EXAKTFILE(P,PARA)` calculates the exact solution `U` in points `P`. Here `P` are not only gridpoints but degrees of freedom.

Chapter 6

Theory

This chapter provides a short introduction to the theory. It serves as a reference for the parameters that can be set in `HelmPole`. The PML method is derived for the simple case of the one dimensional Helmholtz equation. The interpretation of the PML used here is a complex continuation of the solution in the exterior domain. This ansatz can be found for example in the papers by Lassas and Somersalo [7, 6] and Collino and Monk [2]. Next the discretization of the exterior of a bounded convex polygonal two dimensional domain in a ray-like manner is explained. A much more detailed description of the theory can be found in Ref. [9].

6.1 Helmholtz equation

The basic equations we consider are of Helmholtz type. To be more precise two types of equations are considered:

$$\nabla \cdot \left(\frac{1}{\epsilon(\boldsymbol{x})}\nabla u(\boldsymbol{x})\right) + k_0^2 u(\boldsymbol{x}) = 0. \tag{6.1}$$

and

$$\Delta u(\boldsymbol{x}) + k_0^2 \epsilon(\boldsymbol{x}) u(\boldsymbol{x}) = 0. \tag{6.2}$$

These equations hold for $\boldsymbol{x} \in \widetilde{\Omega}$, where $\widetilde{\Omega}$ is in general infinite. In addition these equations have to be completed by boundary conditions on the "interior" boundary Γ_{int} of $\widetilde{\Omega}$ and radiation boundary conditions at infinity. The region of interest is denoted by Ω (see Figure 6.1). We define the "outer" boundary Γ_{ext} to be the part of $\partial\Omega$ that is not part of the "interior" boundary Γ_{int}. On Γ_{ext} u is the sum of the known incoming field u_i and the scattered field u_s. The aim is to obtain a relation between u_s and $\partial_\nu u_s$ on Γ_{ext}. The wavenumber k is defined by $k^2(\boldsymbol{x}) := k_0^2 \epsilon(\boldsymbol{x})$.

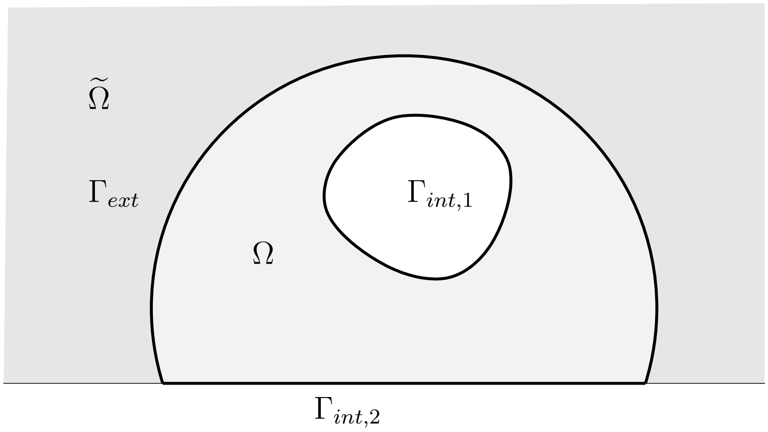

Figure 6.1: Computational domain Ω within the physical domain $\widetilde{\Omega}$. The boundaries of Ω are Γ_{ext}, $\Gamma_{int,1}$, and $\Gamma_{int,2}$. Boundary values are set on $\Gamma_{int,i}$.

Thus restricted to Ω Equation (6.2) reads

$$\begin{aligned} \triangle u(\boldsymbol{x}) + k^2(\boldsymbol{x})u(\boldsymbol{x}) = 0 \quad \text{in } \Omega, \\ u(\boldsymbol{x}) = u_i(\boldsymbol{x}) + u_s(\boldsymbol{x}) \quad \text{on } \partial\Omega, \\ \partial_n u(\boldsymbol{x}) = \partial_n u_i(\boldsymbol{x}) + \partial_n u_s(\boldsymbol{x}) \quad \text{on } \partial\Omega. \end{aligned} \tag{6.3}$$

Here ∂_n denotes the outward normal derivative, with the normal $n := \frac{\boldsymbol{x}}{|\boldsymbol{x}|}$. In the subsequent sections only Equation (6.2) is considered.

6.2 The PML method in 1D

Consider the one dimensional Helmholtz equation on the semi-infinite interval $\widetilde{\Omega} = [-1, \infty)$

$$\begin{aligned} \partial_{xx} u + k^2 u = 0 \\ u(-1) = 1 \\ \partial_\nu u = iku \text{ for } x \to \infty \ . \end{aligned} \tag{6.4}$$

We want to restrict this equation to the interval $[-1, 0]$. Thus the exterior is $(0, \infty)$. The idea of the PML method is to transform the coordinate

$$\overline{x} := \gamma(x) \tag{6.5}$$

where for example

$$\gamma(x) = (1 + \sigma i)x \ . \tag{6.6}$$

As $\gamma(0) = 0$ and $\gamma'(0) = 1 + \sigma i$ the solution of the transformed equation $\tilde{u}_{PML}(x) = u(\gamma(x))$ coincides with u at 0. Assuming that k is constant for $x > 0$, the solution of Equation (6.4) can be extended analytically to the complex plane, hence the normal derivative at 0 is $\partial_\nu u(0) = \partial_\nu \tilde{u}_{PML}(0)/(1+i\sigma)$ and the solution $\tilde{u}_{PML}$ is given by

$$\begin{aligned} \partial_{xx}\tilde{u}_{PML} + k^2(1+i\sigma)^2\tilde{u}_{PML} &= 0 \\ \tilde{u}_{PML}(0) &= u(0) \\ \partial_x\tilde{u}_{PML}(x) &= ik\tilde{u}_{PML}(1+i\sigma) \ . \end{aligned} \tag{6.7}$$

The two fundamental solutions of the PDE in Equation (6.7) are

$$x \mapsto \exp\left(ik(1+i\sigma)x\right) = \exp\left(ikx - k\sigma x\right) \tag{6.8}$$

and

$$x \mapsto \exp\left(-ik(1+i\sigma)x\right) = \exp\left(-ikx + k\sigma x\right) \tag{6.9}$$

Solution (6.9) is exponentially increasing and Solution (6.8) is exponentially decreasing and obeys the radiation condition. To obtain the PML replace (6.7) by

$$\begin{aligned} \partial_{xx}u_{PML} + k^2(1+i\sigma)^2u_{PML} &= 0 \\ u_{PML}(0) &= u(0) \\ u_{PML}(\rho) &= 0 \ . \end{aligned} \tag{6.10}$$

The error introduced by replacing $\tilde{u}_{PML}$ with u_{PML} decreases exponentially with the thickness ρ of the PML.

Next we will discuss the discretization of the exterior using rays. This will lead to a system of ODEs. The derivation of the PML for the one dimensional case can then be extended to this discretization. Details on the PML in the two dimensional case can be found in [4]. For details on the implementation of the PML the reader is referred to [9].

6.3 Discretization of the exterior

Turning back to the two dimensional Helmholtz equation. Suppose the computational domain is a convex polygon, the exterior is then decomposed into a finite number of segments $Q_j^{(x,y)}$ $(j = 1, ..., N)$, as shown in Figure 6.2. The vertices of the polygonal boundary $\partial\Omega$ are connected with infinity by non-intersecting rays. There are different possibilities for constructing these rays, *cf.* [8]. `HelmPole` provides several routines for that, which are explained in Section 5.6.

Each segment $Q_j^{(x,y)}$ is the image of a semi-infinite rectangle $Q_j^{(\xi,\eta)}$ under the mapping $Q_j^{(loc)} : Q_j^{(\xi,\eta)} \rightarrow Q_j^{(x,y)}$. The construction is described in [8]. The local mappings are combined to a global mapping

$$Q : \overline{\Omega_{ext}^{(\xi,\eta)}} \rightarrow \overline{\Omega_{ext}^{(x,y)}}. \tag{6.11}$$

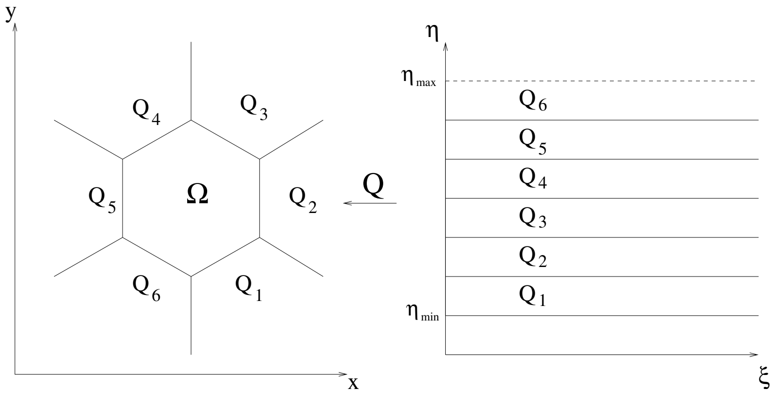

Figure 6.2: Mapping from the semi-infinite rectangular domain to the exterior domain.

Its Jacobian is denoted by J. The exterior problem can be formulated in the ξ, η-coordinate-system. The transformed Helmholtz equation is

$$\nabla_{\xi,\eta}(J^{-1}J^{-T}|J|\nabla_{\xi,\eta})u_s + |J|k^2 u_s = 0 \ . \tag{6.12}$$

The details on the discretization of Equation (6.12) can be found in [9]. We will only give a sketch here.

The ansatz

$$\tilde{u}_s^h(\xi,\eta) = \sum_{j=1}^{N} \tilde{u}_{s,j}(\xi) w_j(\eta), \tag{6.13}$$

with $w_j \in V$, where V is a discrete space, periodic on $[\eta_{min}, \eta_{max}]$ yields a system of second order ODEs, *cf.* [9]

$$A_0 \tilde{\boldsymbol{u}}_s(\xi) + A_1 \partial_\xi \tilde{\boldsymbol{u}}_s(\xi) + A_2 \partial_\xi^2 \tilde{\boldsymbol{u}}_s(\xi) = 0. \tag{6.14}$$

It is shown in [4] that, provided k is constant in each segment, Equation (6.14) can be extended analytically to the complex plane. k being constant in each segment allows to model infinite waveguide structures entering the computational domain. To obtain the PML the transformed equation is replaced by an equation on a finite domain with, *e.g.*, Dirchlet boundary condition.

Chapter 7

Grid Generation

We recommend the use of the triangulation tool `triangle`. `Triangle` needs a description of the geometry as input file (`.poly`-file) and creates files containing the nodes, elements and boundary of a triangulation, respectively. The handling of triangle and the description of the input file is found on the web-site `http://www-2.cs.cmu.edu/~ quake/triangle.html`.

We include several examples of input files for `triangle` in the folder `/trianglefiles/`, *e.g.*, `w2.poly` which is used for the grid generation for the waveguide problem described in section 4.1. The file consists of lists of points, edges, holes, and regional attributes.

We also include an interface to triangle which converts the triangle output files to a single file as input for HelmPole (geometry file, see chapter 5.3). This interface consists of the function `triangle2mat.m`, located in the `/trianglefiles/` directory.

7.1 Conversion interface: TRIANGLE2MAT

Syntax

`[P,E,T]=TRIANGLE2MAT(NAME)`

Arguments

- `NAME` Name of the output-files of triangle, located in the directory `/HelmPole/trianglefiles/`, without the file extensions `.1.node, .1.ele, .1.poly`.

Description

This function reads the files `/HelmPole/trianglefiles/ name.1.node, name.1.poly,name.1.ele` and writes a geometry-file `/HelmPole/geo/name.mat`.

Please note the following convention for edge identifiers: for boundaries to the exterior: $id <= 100$; for interior boundaries: $100 < id <= 200$; for inner edges (not on a boundary): $id > 200$.

Chapter 8

License and Copyright

Referencing

We kindly ask you to reference the program and its authors in any publication for which you used HelmPole. The preferred citation is this manual:
S. Burger, R. Klose, A. Schädle, and L. Zschiedrich, "HelmPole - A finite element solver for scattering problems on unbounded domains: Implementation based on PML," ZIB-Report 03-38, Zuse Institute Berlin, (2003).

Or, in BibTeX format:

```
@techreport{HelmPole2003,
author      = { S. Burger and R. Klose and A. Sch\"adle
                and L. Zschiedrich },
title       = { HelmPole - A finite element solver for scattering
                problems on unbounded domains:
```

```
                Implementation based on PML },
institution = { Zuse Institute Berlin },
number      = { 03-38 },
year        = { 2003 } }.
```

Updates

Please note that the program `HelmPole` is intended to be constantly updated. Updated versions of the code and of this manual will be available from the homepage of the *Computational Microwave Technology* group at the Konrad-Zuse Center for Information Technology, Berlin, `http://www.zib.de/Commic`.

Acknowledgements

We thank P. Deuflhard and F. Schmidt for support and discussions. We acknowledge support by the initiative *DFG Research Center "Mathematics for key technologies"* of the Deutsche Forschungsgemeinschaft (German Research Foundation) and by the German Federal Ministry of Education and Research, BMBF, under contract no. 13N8252 (*"HiPhoCs"*).

Bibliography

[1] J.-P. Bérenger. A perfectly matched layer for the absorption of electromagnetic waves. *J. Comput. Phys.*, 114(2):185–200, 1994.

[2] F. Collino and P. Monk. The perfectly matched layer in curvilinear coordinates. *SIAM J. Sci. Comput.*, 1998.

[3] M. Hammer. Resonant coupling of dielectric optical waveguides via rectangular microcavities: the coupled guided mode perspective. *Optics Communications*, 214(1-6):155–170, 2002.

[4] T. Hohage, F. Schmidt, and L. Zschiedrich. Solving time-harmonic scattering problems based on the pole condition: Convergence of the PML method. Preprint 01-23, Konrad-Zuse-Zentrum (ZIB), 2001.

[5] F. Ihlenburg. *Finite Element Analysis of Acoustic Scattering*, volume 132 of *Applied Mathematical Sciences*. Springer-Verlag, New York, 1998.

[6] M. Lassas, J. Liukkonen, and E. Somersalo. Complex Riemannian metric and absorbing boundary condition. *J. Math. Pures Appl.*, 80(7):739–768, 2001.

[7] M. Lassas and E. Somersalo. On the existence and convergence of the solution of PML equations. *Computing*, 60(3):229–241, 1998.

[8] F. Schmidt. *A New Approach to Coupled Interior-Exterior Helmholtz-Type Problems: Theory and Algorithms*. Habilitationsschrift, Konrad-Zuse-Zentrum, Berlin, 2002.

[9] F. Schmidt, L. Zschiedrich, R. Klose, and A. Schädle. A new finite element realization of the perfectly matched layer method for helmholtz scattering problems on polygonal shaped domains in 2d. to be submitted, 2003.

[10] H.-G. Unger. *Optische Nachrichtentechnik, Teil 1 Optische Wellenleiter*. Studientexte Elektrotechnik. Hüthig, 2. edition, 1990.

[11] L. Zschiedrich. *Transparent boundary conditions for time-harmonic scattering problems and time-dependent Schroedinger equations.* PhD thesis, Fachbereich Mathematik und Informatik, FU Berlin, in preparation 2003.